Trust-based personal information management in SOA

Guillaume Feuillade, Andreas Herzig, Kramdi Seifeddine

Trust-based Personal Information Management in SOA

Guillaume Feuillade, Andreas Herzig and Kramdi Seifeddine
IRIT, Universite Paul Sabatier, Toulouse, France
{*guillaume.feuillade, andreas.herzig, kramdi.seifeddine*}*@irit.fr*

Keywords: Service Oriented Architecture, SOA, Trust Management, Epistemic Logic.

Abstract: Service Oriented Architecture (SOA) enables cooperation in an open and highly concurrent context. In this paper, we investigate the management of personal information by an SOA service consumer while invoking composed services, where we will study the balance between quality of service (that works better when provided with our personal data) and the consumer's data access policy.We present a service architecture that is based on an open epistemic multi-agent. We describe a logic-based trust module that a service consumer can use to assess and explain his trust toward composed services (which are perceived as composed actions executed by a group of agents in the system).We then illustrate our solution in a case study involving a professional social network.

1 INTRODUCTION

In this paper, we investigate the management of personal information by an SOA service consumer while invoking composed services. Generally speaking, managing personal information relates to different aspects of data management, such as storage, persistence and querying. In this work, we focus on privacy and confidentiality in a business to user (b2u) application and critical information dissemination in business to business (b2b) solutions. Where we study the balance between quality of service and the consumer's data access policy. We present a service architecture based on an open multi-agent system where agents provide and invoke (composed-)services in order to achieve their goals. We describe a logic-based trust module that a service consumer can use to assess his trust toward composed services. Using an abduction mechanism, we provide the service consumer with a synthesized view of his beliefs about the current state of the multi-agent system. This view is coupled with an answer to the question: why should I trust or not this composition? We then illustrate our solution in a case study about a *personal information management system* (PIMS) involving a professional social network. We discuss how the CEO of a start-up company, collects information about Alice, a potential recruit, while preserving his personal information, using composed services provided by the members of the network including Alice herself.

2 THE PIMS NETWORK ARCHITECTURE

In a service based application, functions are encapsulated within what is called a *service*, where a set of standards that describe the interaction protocol can be used to interact with the service (Holley and Arsanjani, 2010). A *service consumer* asks a *service provider* to access some functionality that is provided by one of his services. In order to provide complex services, the base services can be composed for behaving together as a composed service.

Our goal is to put the management of personal information in the center of SOA design. To achieve this, we present a framework that is based on the concept of *personal information management system*, abbreviated PIMS [1]. A PIMS is a software entity that is associated to an individual or an organization. Via his PIMS, an agent can provide and summon services in the network, while managing the access of his personal information. A PIMS is made up of the following three major components.

Database. The PIMS database contains personal information about the owner of the PIMS. This data includes:

- *Categorization data*: it classifies the semantics of the user's personal information using an ontology;

[1] https://en.e-citiz.com/innovation/pimi-how-do-we-take-take-of-you-personal-information.

- *Key entries* or values: this element corresponds to the actual data in the PIMS;
- *Metadata:* this corresponds to non-functional data, like an access policy that can be specified using standards like XACML or EPAL (Khare, 2006).

Let $D = \{d_1, d_2, \ldots, d_n\}$ be the set of relevant data labels.

Services Repository. In order to help the user to execute services when playing the role of the service provider, this component contains the description of a set of services that other PIMS of the network can invoke. Such services provides at the same time access to the owner's personal information (upon request).

Query Interface. A query interface allow the user to specify a goal to be achieved by composition of the available services. a service composition components uses then this query to provide a list of service compositions that achieve the goal. These services are then annotated by the PIMS's trust module using informations that can help the user to choose witch one to execute(prediction of quality of service, trust evaluation of the composition,etc.).

3 ILLUSTRATION: CASE STUDY

We present a case study about the process of retrieving information in a professional social network.[2]

Suppose Bob is a member of a social network who wants to gather information about Alice: more precisely Bob would like to retrieve her curriculum vitæ (CV), which includes the name of the owner of the CV (identification reference), a proof of level in English and a recent certified recommandation letter (in our case, form Mr. Duval, one of Alice's exemployer). Suppose that Bob also specifies that he would like to stay anonymous while retrieving Alice's CV.

The service composition component provides Bob with the following compositions to achieve his goal:

`Alice.cv()` :where Alice offers to access a copy of her CV. Figure 1 shows that her CV includes her identification, her Cambridge proficiency in English

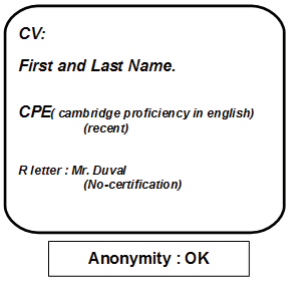

Figure 1: Alice's CV

mark (that she passed recently) and a recommendation letter from Mr. Duval, that is not certified. This service does not ask any identification to be executed.

`Pro.cv()`: provided by a professional company that offers to the network users access to information about other users that it has gathered from different sources. `Pro.cv()` offers the same services as `Alice.cv()` ,with the exception that `Pro.cv()` actually certifies M. Duval's letter. Moreover, `Pro.cv()` asks for an access fee of 10 euros, while not asking Bob for any identification.

`Composition1()`: the user can construct the CV by himself, using services that are available in the network. This composition needs Alice's name (retrieved by querying her PIMS) and uses it to collect Alice's score at the TOEFL exam; it then contacts Mr. Duval (whose contacts are provided by Alice) and directly asks him for a recommendation letter. Mr. Duval's recommendation letter will be certified. A drawback of this composition is that Bob may retrieve an outdated TOEFL score. M. Duval requires Bob's identity in order to provide the recommendation letter.

Then the trust module enters the scene in order to analyze the three proposed compositions. The trust module:

1. constructs a model of Bob's knowledge about the network;
2. analyzes the compositions using a formalization of the concept of trust to be presented in Section 5;
3. asserts the trust value and generates a set of explanations about the trust status.

In our example, the trust module generates the following output.

- For `Alice.cv()`: trust towards this execution fails, since the recommendation letter is not certified.

[2]We define a professional social network as a network of PIMS where the users share their personal information in order to look for job opportunities. This means that each PIMS provides a way to access the network members information via services.

- For `Composition1()`: trust towards this execution will also fail, because the privacy requirement is not met.
- For `Pro.cv()`: this composition satisfies all the requirements for a positive trust value, but Bob is notified that there is a fee.

Based on the annotations that the trust module has provided to him, Bob may now decide which service he wants to execute. He may decide to retract his privacy concerns (for example because he considers that M. Duval can be trusted about keeping secret personal information) and choose `Composition1()`. Bob may as well accept to pay the transaction fees and pursue the `Pro.cv()` service execution.

4 THE TRUST MODULE

As described above, our trust module takes into account the model that an agent constructs about the network, in order to propose a set of annotations toward possible action compositions. Interaction in the PIMS network takes place according to the following steps:

1. The user enters a query, describing the functional properties of the service that he wants to invoke.
2. The composition module uses the query to propose a set of service compositions that are likely to resolve the query.
3. Based on the user's knowledge about the PIMS network, his trust module adds trust-related annotations to the compositions so that an informed choice can be made.
4. The user chooses the composition to be executed.

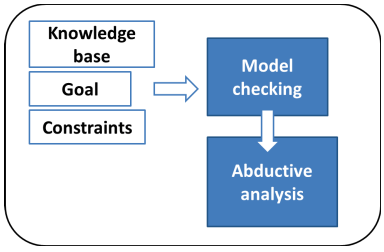

Figure 2: The Trust module.

So the trust module needs to assess the current situation and provide information about the composition. In order to do so, we define the trust module (as illustrated in Figure 2) by the following components.

Knowledge Base. In order to analyze the current situation, the PIMS maintains a model of the current state of the PIMS network. This model includes information about the other PIMS (their beliefs about the network and their goals and preferences) and properties of the network (resource availability, quality of service of the different services that are available, etc.). Such information may be deduced from past interaction, knowledge of the user about the network (including his own services for example) and information provided by monitoring services. We suppose that the knowledge base is written in the logic that we are going to present in the next section.

Goal and Constraints. When initiating a new interaction, the trust component receives the description of the query to be resolved (his goal) and the preferences of the user (a set of constraints) such as Bob's anonymity goal. This information enables the module to build beliefs about service compositions that resolve the query while guaranteeing the preferences that were specified.

Model Checking. Trust is described in terms of a belief about the conditions under which the user can trust a composition to achieve his goal. Since our trust module is based on a logical formalization, this is described by a logical formula, and asserting trust is reduced to reasoning about its components. At this step, we should be able to answer the question whether the user should trust or not the current composition to achieve the goal while respecting the user's preferences.

Abductive Analysis. Once (dis)trust is asserted, an abductive procedure associates to the current trust value an explanation justifying it. This explanation is displayed to the user in order to guide him in deciding which composition to choose.

5 THE LOGIC

Our logic is a fairly standard multimodal logic of action, belief, and choice that we call ABC and that combines the logic for trust of (Herzig et al., 2010) with the logic of assignments of (Balbiani et al., 2013). The logic ABC is decidable.

There is a countable set of *propositional variables* $\mathbb{P} = \{p, q, \ldots\}$ and a finite set of *agent names* $\mathbb{I} = \{i, j, k, \ldots\}$. In the epistemic dimension of the language, we have modal operators of belief $\mathtt{Bel}_i$ and choice $\mathtt{Ch}_i$, one per agent $i \in \mathbb{I}$. $\mathtt{Bel}_i\varphi$ reads "i believes that φ", and $\mathtt{Ch}_i\varphi$ reads "i chooses that φ", or "i prefers that φ", where φ is a formula.

The dynamic dimension of the language is based on *assignments*. An assignment is of the form either p^+ or p^-, where p is a propositional variable from $\mathbb{P}$; p^+ makes p true and p^- makes p false. An *authored*

assignment is a couple of the form either $\langle i, p^+\rangle$ or $\langle i, p^-\rangle$, where i is an agent from $\mathbb{I}$ and p is a variable from $\mathbb{P}$. The intuition for the former is that i sets the variable p to true; for the latter it is that i sets p to false. For ease of notation we write $i{:}{+}p$ and $i{:}{-}p$ instead of $\langle i, p^+\rangle$ and $\langle i, p^-\rangle$. An *atomic action* is a finite set of authored assignments. Given an atomic action δ and an agent $i \in \mathbb{I}$, we define *i's part of δ* as:

$$\delta|_i = \{i{:}{+}p \mid i{:}{+}p \in \delta\} \cup \{i{:}{-}p \mid i{:}{+}p \in \delta\}$$

The set of all atomic actions is noted Δ.

Beyond the modal operators $\mathtt{Bel}_i$ and $\mathtt{Ch}_i$, our language has two dynamic modal operators $\langle . \rangle$ and $\langle\langle . \rangle\rangle$ the first of which is from Propositional Dynamic Logic PDL. These operators have complex actions as arguments. The formula $\langle \pi \rangle \varphi$ reads "the action π is executable and φ is true afterwards". In contrast, for $\langle\langle \pi \rangle\rangle \varphi$ reads "π is executed and φ is true afterwards". The latter implies the former: when π is executed then π clearly should be executable. It is also clear that the other way round, executability should not imply execution.

Therefore $\langle i{:}{+}p \rangle \top$ reads "i is able to set p to true", while $\langle\langle i{:}{+}p \rangle\rangle \top$ reads "i is going to set p to true". The formula $\langle\langle i{:}{+}p, j{:}{-}q \rangle\rangle \varphi$ expresses that the agents i and j are going to assign the value 'true' to the propositional variable p and 'false' to q, and that afterwards φ will be true; and $\mathtt{Bel}_k \langle\langle i{:}{+}p, j{:}{-}q \rangle\rangle \varphi$ expresses that agent k believes that this is going to happen. As the reader may have noticed, we drop the set parentheses around the atomic assignments in formulas such as $\langle i{:}{+}p \rangle \top$, $\langle\langle i{:}{+}p \rangle\rangle \top$ and $\langle\langle i{:}{+}p, j{:}{-}q \rangle\rangle \varphi$.

Formally, we define by induction the set of *actions* (programs) $\mathtt{Prog}$ and the set of *well-formed formulas* $\mathtt{Fml}$ of ABC logic.

$$\pi ::= \delta \mid \mathtt{skip} \mid \mathtt{fail} \mid \pi;\pi \mid \mathtt{if}\ \psi\ \mathtt{then}\ \pi\ \mathtt{else}\ \pi$$

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{Bel}_i\varphi \mid \mathtt{Ch}_i\varphi \mid \langle \pi \rangle \varphi \mid \langle\langle \pi \rangle\rangle \varphi$$

where p ranges over $\mathbb{P}$, i over $\mathbb{I}$ and δ over Δ. Here is an example of a complex action. Let L mean that the light is on. Then $\mathtt{if}\ L\ \mathtt{then}\ j{:}{-}L\ \mathtt{else}\ j{:}{+}L$ describes j's action of toggling the light switch.

We use the standard abbreviations for $\vee$ and $\rightarrow$. Moreover, $\bot$ abbreviates $\neg\top$ and $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$.

The formulas of our language have a semantics is in terms of Kripke models and model updates, as customary in dynamic epistemic logics (van Ditmarsch et al., 2007). We do not go into the details of the semantics here (see (Herzig et al., 2013)) and instead rely on the reader's intuitions. Here are some examples of validities:

- $\langle\langle \pi \rangle\rangle \top \rightarrow \langle \pi \rangle \top$ (do implies can)
- $\mathtt{Bel}_i\varphi \rightarrow \mathtt{Ch}_i\varphi$ (realistic choice)
- $\mathtt{Ch}_i\varphi \rightarrow \mathtt{Bel}_i\mathtt{Ch}_i\varphi$ (positive introspection)
- $\neg\mathtt{Ch}_i\varphi \rightarrow \mathtt{Bel}_i\neg\mathtt{Ch}_i\varphi$ (negative introspection)
- $\langle\langle j{:}a \rangle\rangle \top \rightarrow \mathtt{Ch}_j \langle\langle j{:}a \rangle\rangle \top$ (intentional action)

It is decidable whether a formula is true in a given ABC model. It is also decidable whether a formula is satisfiable in the set of ABC models.

5.1 Formalizing Trust

We now turn to a formalization of a theory of trust in complex actions in ABC logic. The trust theory basically extends Castelfranchi and Falcone's.

Among the different theories of trust, the cognitive theory of Castelfranchi and Falcone, henceforth abbreviated C&F, is probably most prominent (Castelfranchi and Falcone, 1998; Falcone and Castelfranchi, 2001).

According to C&F, the trust relation involves a truster i, a trustee j, an action a that is performed by j and a goal φ of i. They defined the predicate $\mathtt{Trust}$ as a goal together with a particular configuration of beliefs of the trustee. Precisely, i trusts j to do a in order to achieve φ if and only if i has the goal that φ and i believes that:

1. j is capable to perform a,
2. j is willing to perform a,
3. j has the power to achieve φ by doing a.

C&F distinguish external from internal conditions in trust assessment: j's capability to perform a is an external condition, while j's willingness to perform a is an internal condition (being about the trustee's mental state). Finally, j's power to achieve φ by doing a relates internal and external conditions: if j performs a then φ will result. Observe that in the power condition, the result is conditioned by the execution of a; therefore the power condition is independent from the capability condition. In particular, j may well have the power to achieve φ without being capable to perform a: for example, right now I have the power to lift a weight of 50kg, but I am not capable to do this because there is no such weight at hand.

We follow Jones who argued that the core notion of trust need not involve a goal of the truster (Jones and Firozabadi, 2001; Jones, 2002) and consider a simplified version of C&F's definition in terms of a truster, a trustee, an action of the trustee, and an expected outcome of that action.

Complex action expressions involve multiple agents that occur in the action expressions. We therefore need not identify the trustee as a separate argument of the trust predicate. Our official definition of

the trust predicate then becomes:

$$\texttt{Trust}(i,\pi,\varphi) \stackrel{\text{def}}{=} \texttt{Bel}_i(\texttt{CExt}(\pi) \wedge \texttt{CInt}(\pi) \wedge \texttt{Res}(\pi,\varphi))$$

where $\texttt{Bel}_i$ is a modal operator of belief and where $\texttt{CExt}(\pi)$, $\texttt{CInt}(\pi)$, and $\texttt{Res}(\pi,\varphi)$ respectively correspond to items 1, 2 and 3 in C&F's definition. CExt and CInt stand for the external and the internal conditions in trust assessment. We prefer these terms and notations because they better generalize to complex actions. We also call CExt the executability condition and CInt the execution condition. The modal operators $\texttt{Bel}_i$ are already primitives of ABC logic. It remains to define the other components of trust in ABC logic:

$$\begin{aligned}\texttt{CExt}(\pi) &\stackrel{\text{def}}{=} \langle\pi\rangle\top\\ \texttt{CInt}(\pi) &\stackrel{\text{def}}{=} \langle\pi\rangle\top \rightarrow \langle\langle\pi\rangle\rangle\top\\ \texttt{Res}(\pi,\varphi) &\stackrel{\text{def}}{=} [\pi]\varphi\end{aligned}$$

The definition of the internal condition says that if π is executable then π is going to happen. This is actually a bit weaker than C&F's willingness condition. To see this, consider the case where π is an atomic action a of some agent j, written $j{:}a$. If j cannot perform a, i.e., when the external condition fails to hold, then the internal condition is trivially true. There is however no harm here: as $\texttt{CExt}(j{:}a)$ is false, the trust predicate will be false anyway in that case. In the case where the external condition holds the internal condition reduces to truth of $\langle\langle j{:}a\rangle\rangle\top$, and as we have seen in Section 5, $\langle\langle j{:}a\rangle\rangle\top$ implies $\texttt{Ch}_j\langle\langle j{:}a\rangle\rangle\top$, i.e., in that case j indeed has the intention to perform a.

Given the above definitions, by means of valid equivalences of ABC logic we obtain:

$$\begin{aligned}\texttt{Trust}(i,\pi,\varphi) &\leftrightarrow \texttt{Bel}_i(\langle\langle\pi\rangle\rangle\top \wedge (\langle\langle\pi\rangle\rangle\top \rightarrow \langle\pi\rangle\top) \wedge [\pi]\varphi)\\ &\leftrightarrow \texttt{Bel}_i(\langle\langle\pi\rangle\rangle\top \wedge \langle\pi\rangle\top \wedge [\pi]\varphi)\\ &\leftrightarrow \texttt{Bel}_i(\langle\langle\pi\rangle\rangle\top \wedge [\pi]\varphi)\end{aligned}$$

In words, i's trust that the complex action π is going to be performed and produces φ reduces to a belief of i that π is going to occur and that φ is among the effects of π.

6 CASE STUDY IN OUR LOGIC

We now provide an analysis of the elements of the PIMS case study in ABC logic. First, the set of agents $\mathbb{I} = \{i, j, k, \dots\}$ should be the set of PIMS.

Second, the propositional language should encode the agents' knowledge about the other agents' personal data. For every relevant data label d in the set $D = \{d_1, d_2, \dots, d_n\}$ (as introduced in Section 2) and agents i and j, we introduce a propositional variable p_{ijd} expressing that i knows the value of the data label d of agent j. Therefore the set of propositional variables is an extension of the set of knowledge variables: we have that $p_{ijd} \in \mathbb{P}$ for every $i, j \in \mathbb{I}$ and $d \in D$ We should guarantee that positive and negative introspection hold for these variables, i.e., we expect $p_{ijd} \rightarrow \texttt{Bel}_i p_{ijd}$ and $\neg p_{ijd} \rightarrow \texttt{Bel}_i \neg p_{ijd}$ to hold for every propositional variable p_{ijd}.[3] We assume that agents are sincere when they inform about a name: an agent i can inform an agent j of the name of the agent k only if i knows j's name. We therefore expect $\langle i{:}{+}p_{\text{j k name}}\rangle\top \rightarrow p_{\text{i k name}}$ to hold.

Using this formalization, Bob's query can be translated into a formula expressing the knowledge about Alice he would like to have:

$$p_{\text{bob alice name}} \wedge (p_{\text{bob alice toefl}} \vee p_{\text{bob alice pge}}) \wedge \wedge (p_{\text{bob alice } \mathit{letter}_{\mathit{duval}}})$$

Anonymity means that α will not make Bob's name available to whom ignored it before:

$$\bigwedge_{a \in \mathbb{I}} \neg p_{\text{a bob name}} \rightarrow \neg\langle\langle\alpha\rangle\rangle p_{\text{a bob name}}$$

where α is one of the three possible compositions. We also assume the use of a set of propositional variables to represent Bob preferences (like having Mr. Duval letter certified by him (certif), or obtaining an old english test (old), likewise, feepaid is a propositional variable related to the cost of (pro.cv()).

The complex actions that are proposed by the service composition module are the following:

$\texttt{Alice.cv}() : \{(\mathit{alice}{:}{+}p_{\text{bob alice name}}),$
$\quad (\mathit{alice}{:}{+}p_{\text{bob alice pge}}),$
$\quad (\mathit{alice}{:}{+}p_{\text{bob alice } \mathit{letter}_{\mathit{duval}}})\}$
$\texttt{Pro.cv}() : \texttt{if feepaid then } \{\mathit{pro}{:}{+}p_{\text{bob alice name}},$
$\quad (\{\mathit{pro}{:}{+}p_{\text{bob alice pge}}),$
$\quad (\{\mathit{pro}{:}{+}p_{\text{bob alice } \mathit{letter}_{\mathit{duval}}})\} \texttt{ else fail}$
$\texttt{Composition1}() :$
$\quad \{(\{\mathit{alice}{:}{+}p_{\text{bob alice name}}), (\{\mathit{alice}{:}{+}p_{\text{bob duval name}})\};$
$\quad \{(\mathit{toefl}{:}{+}p_{\text{bob alice toefl}})\};$
$\quad \texttt{if } p_{\text{bob duval name}} \texttt{ then}$
$\quad\quad \{(\mathit{duval}{:}{+}p_{\text{bob alice duvalrecommandation}}),$
$\quad\quad (\mathit{bob}{:}{+}p_{\text{duval bob name}})\} \texttt{ else skip}$

The trust analysis process introduced in section 3 is reduced to the satisfiability checking of the trust predicates, associated to the composed actions and the goal of Bob, the abduction mechanism allows us to

[3] Note that while $\neg\texttt{Bel}_i\neg p_{ijd} \rightarrow p_{ijd}$ follows from that, the formula $\texttt{Bel}_i p_{ijd} \rightarrow p_{ijd}$ does not. The reason is that the modal operators $\texttt{Bel}_i$ do not obey the D axiom $\texttt{Bel}_i p_{ijd} \rightarrow \neg\texttt{Bel}_i\neg p_{ijd}$.

associate, to the trust decision, a possible explanation to present to the user, where:

For `Alice.cv()`: trust towards this execution fails, since the recommendation letter is not certified (which can be remarked by observing that alice is the one who assigned $p_{\text{bob duval name}}$ to true).

For `Composition1()`: trust towards this execution will also fail, because the privacy requirement is not met (due to the fact that bob has to put $p_{\text{duval bob name}}$ to true during the execution).

For `Pro.cv()`: this composition satisfies all the requirements for a positive trust value, but Bob is notified that there is a fee (by the necessity to have feepaid true).

7 CONCLUSIONS AND DISCUSSION

In this paper, we have presented a case study related to social professional networks using a SOA design, where we have demonstrated, how a symbolic approach can model interactions, particularly how an agent can manage his trust in a composed service. We believe that our work can be used as a ground both for theoretical and conceptual study of the application of formal methods to service-oriented design.

As for the theoretical part, our logic can be extended in order to handle more complex interactions, the most relevant way to do so would be to extend the set of action constructors in a way that preserves the decidability of the logic.

A conceptual extension of our work can correspond to the study of how our logic can be used to represent concepts involved in service oriented design. Such concepts can be related to resource management, interaction protocols and access policies.

We hope that our work can be used as a proof that formal verification brings an interesting perspective over service-oriented design, where formal specification can be used both to prove properties of some SOA application, while defining a blueprint to implement intelligent agents interacting in an open environment.

ACKNOWLEDGEMENTS

We would like to thank one of the two anonymous reviewers of ICAART for his thorough reading of the submitted paper. Our work was done within the framework of the ANR VERSO project PIMI .

REFERENCES

Balbiani, P., Herzig, A., and Troquard, N. (2013). Dynamic logic of propositional assignments: A well-behaved variant of PDL. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 143–152. IEEE Computer Society.

Castelfranchi, C. and Falcone, R. (1998). Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In *Proceedings of the Third International Conference on Multiagent Systems (ICMAS'98)*, pages 72–79.

Falcone, R. and Castelfranchi, C. (2001). Social trust: A cognitive approach. In Castelfranchi, C. and Tan, Y. H., editors, *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer.

Herzig, A., Lorini, E., Hübner, J. F., and Vercouter, L. (2010). A logic of trust and reputation. *Logic Journal of the IGPL*, 18(1):214–244. Special Issue "Normative Multiagent Systems".

Herzig, A., Lorini, E., and Moisan, F. (2013). A simple logic of trust based on propositional assignments. In Paglieri, F., Tummolini, L., Falcone, R., and Miceli, M., editors, *The Goals of Cognition. Essays in Honor of Cristiano Castelfranchi*, pages 407–419. College Publications, London.

Holley, K. E. and Arsanjani, A. (2010). *100 SOA Questions: Asked and Answered*. Pearson Education.

Jones, A. J. I. (2002). On the concept of trust. *Decision Support Systems*, 33(3):225–232.

Jones, A. J. I. and Firozabadi, B. (2001). On the characterisation of a trusting agent - aspects of a formal approach. In Castelfranchi, C. and Tan, Y. H., editors, *Trust and Deception in Virtual Societies*, pages 157–168. Kluwer.

Khare, R. (2006). Microformats: the next (small) thing on the semantic web? *Internet Computing, IEEE*, 10(1):68–75.

van Ditmarsch, H. P., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*. Kluwer Academic Publishers.